A draft proposal for a standard for the coding of machine readable sources

Thaller, Manfred

Manfred Thaller*

A Draft Proposal for a Standard for the Coding of Machine Readable Sources**

<< "Machine readable sources" we call all representations of a source on an electronic computer which are used by source-oriented data processing. So we include data, which contain significant portions of text along with coded information - as available in the data banks of demographic research (in its broadest meaning); we include at the same time material which consists almost completely of text, but of very small units of it, being heavily structured - material, that is, as it exists in the data bases of microanalytical research; and we also include, finally, more or less uncoded representations of continuous machine readable texts - as usually resulting from computer aided editing techniques. >>

Introduction

This paper has been written to support three aims:

a) Presenting a proposal for a general standard for the description and definition of input formats to be used in the handling of machine readable historical material.
b) Submitting a proposal for a general definition of a machine readable format in which such data can be described formally for an interchange program.
c) Finally, this proposal is at the same time a working paper for the ongoing work on the software system to be known as CLIO/C, being currently developed jointly by the Max-Planck-Institut für Geschichte and a number of other institutions.

So the following proposal is linked practically to some ongoing work; it is not considered to be biased in favor of any existing software, being, quite on the contrary, a description of existing solutions to more general problems. As such it is very down to earth; it has an Utopian component, however, as we think it to be a small contribution to what we consider one of the major developments to be made to make secondary analysis of machine readable data collected by source oriented data processing practicable. An interface between historical software systems could be build like this: Software system "A" creates out of its knowledge about the data it administers and analyses, as stored in its data dictionary, a neutral machine readable description of the data that are stored within a given data set and writes it to a machine readable medium. Software system "B" reads this description and is therefore able to convert the data originally processed by software system "A" into a format it can handle itself.

Initial Considerations and Definitions

Any standard of machine readable data used in the historical disciplines, has to be operative at five levels:

---

* Address all communications to: Manfred Thaller, Max-Planck-Institut für Geschichte, Hermann-Föge-Weg 11, D 3400 Göttingen.

** Originally submitted to the international workshop on standardization and exchange of machine readable data in the historical disciplines, held at the University of Graz, May 29th through June 1st, 1986

a) Preparing conventions for the description of characters not in the character set of some of the devices used, as e.g. letters with diacritical marks ("ä", "á" etc.) or letters which are missing in the local alphabet ("L, T"). At the same time, this level has to take care of the definition of portions of the text, which have a technical meaning for the presentation of the text, not a meaning for the description of its contents, as e.g. the instruction telling the printer to change fonts.

b) Providing means for a description of all conventions used for "special text sorts". A "special text" within this document is a portion of letters and/or other characters, which have to be understood according to some fixed syntactical rule to divide their content between a number of conceptual variables, making up some well defined item of information, which can be handled by a suitable subroutine library, as e.g. calendar dates.

c) Presenting a way to describe the logical structure of the items of information making up a historical source: defining, that is, at the most elementary level, at which point the data pertaining to one person end and the ones related to a second one start.

d) Proposing an ideal type for complexes of information, which occur in historical data independently of their place and/or time of origin and can therefore be compared between those sources most easily: "persons", "pieces of land", "change of location" etc.

e) Submitting, finally, prototypes for the treatment of the most frequent types of historical source material, as, e.g. "testaments", "tax registers" and so on.

Any such effort should, furthermore, take two restrictions into consideration:

a) There are a number of software systems in existence, which have been used to handle historical sources. Any standard has therefore to be written independently of the conventions of any such system. It is obvious, that some of the software systems used today are not able to handle structures supported by others: the transformation of machine readable data between two systems has therefore to decide what is going to happen to data treated by the software system the come from in a specific way not available within the system they shall be used with further on. To solve this problem, we attempt in this paper to describe the properties of source material in a general form and assume, that any translation process between two software systems will translate portions of information, that can not be handled by the software system that shall receive the translated data, into a leftover category, that can be handled by the system in question.

b) While the five levels of standardization having been described just above, are clearly distinguishable logically, during the empirical process of data preparation for machine supported analysis, the are by no means distinguished as clearly. For example: while in many historical data sets, that came into existence as a result of computer supported publishing, a lot of information that is present deals simply with formal properties of parts of the text ("print the following string in italics"), when such information is translated into another system which shall analyze those data, such formal properties can carry semantic meaning - say everything in italics happens to be a personal name.

Throughout this document we use a number of phrases and symbols with a specific restricted meaning. These are:

"Software System": A set of computer programs, that has been written to perform a specific task, provided the data submitted to it follow specific syntactical rules, known as its "input conventions".

"Machine Readable Source": Any set of information extracted from a historical source - (which can also contain the fully transcribed text) - according to some rules, which form a subset of the input conventions of some software system, known as the input format of said source.

"Source" or "Source System": The software system which provided the input conventions according to which a given machine readable source, that shall be converted for further processing by another software system, has been prepared.

"Target" or "Target System": The software system, which shall be used for the further processing of a machine readable source stemming from some source system.

"Record": The smallest amount of information, which can be read from a machine readable source, until some physical/logical mark signalling the end of a unit of information is being encountered. (Typically a line, limited by a Carriage Return/Line feed sequence. Also a fixed length portion of text read from a tape.)

What to describe about a machine readable source?

1 Characters and Printing Instructions

On this level, the following qualities of a text are of interest:

1.1 How many bytes have been used to code an input character?

The following decisions are known regarding this question:

- One byte represents one character. Composite symbols - e.g. ":a" for "a" - occur, are however, always treated as a sequence of more elementary characters.
- Another fixed number of bytes represents always precisely one character. (E.g. the two byte or four byte codes used for the transcription of certain Asian scripts, particularly the Chinese one.) Special symbols may be expressed by one byte codes (e.g. a blank for word separation).
- During input a standard one-byte character set is used, as e.g. the standard ASCII set. There exists however, a convention, which allows to separate the input text into a sequence of multicharacter symbols, which for all purposes are treated as exactly one character. (E.g. a solution for cuneiform letters, where words are separated into a set of cuneiform images by hyphens, any image being made up completely of upper case characters being considered an ideographic - rather than syllabic - entry.)

Recommendations: Any description of a source should contain a paragraph defining which of these three solutions has been chosen. If there are a few byte combinations, which are different in length from the remaining ones, a list of them shall be part of the data description. If a basically one-byte-one-character data set is interspersed with ideographic items, that is, with items which represent precisely one meaning, which is not bound to be inflected, changed by adding suffixes, prefixes or the like, a full list of all such items shall accompany the description of the machine readable source.

1.2 How are the individual characters be mapped into the byte **(or** set **of *bytes)?***

While code definitions like ASCII or EBCDIC seem to be obvious, this is not always the case. Particularly EBCDIC contains **a** large number of potential code values, which are not used in standard EBCDIC but available at many sites for characters of importance for national alphabets (or simply traditionally available at a given site). In ASCII certain characters are usually assigned different graphics between the European countries to represent the national alphabets. With microcomputers the "extended ASCII" of IBM, using for national characters and other special characters the code values from 128 through 255, has become more or less a standard for IBM and compatible computers. There exist a large number of modifications to that standard, however, and with some other pretty popular brands of small computers codes are far from standard.

Recommendations: It is therefore recommended, that every machine readable data set has, as part of its documentation, a complete list of all possible code values, together with their graphic representation when used with the printing and/or display facilities of the installation, where the data have been created.

1.3 How are multi character symbols coded, which logically form one character only?

Independent of the basic decision about how many bytes are used to depict one character, most of the data sets taken from historical source material contain certain "composite symbols", that is, symbols which form logically just one character, are described, however, by more than one character.

The most important cases are:

- Two or more characters being treated as one character for some purposes: "sch" for sorting, or the like.
- One character signalling, that the following character is **a** diacritic for the next one: "\"**a**" for "**ä**", "\'**a**" for "**á**" and so on. (In rare cases this construction is reverted, i.e., the character to be modified, stands in front of the diacritic, as in "^**a***" for "**ã**", "**a**' " for "**á**" and so on.)
- One character signalling, that a following character shall be modified by a sequence of numbers following, a sequence, that is, which is either of fixed length, or limited by the next space encountered. E.g.: "&a23" for "á", "&a24" for "à" and so on. (An example for the order "signalling character - modifying code - modified character" is not known to the author, theoretically possible, however.)
- In both the last cases the character announcing the special treatment of the following character can either have a general definition - "when a backslash - '\' - is encountered, always print the character immediately following it over the character following next" - or a special one: "treat the sequence "\' " by printing "' " over whatever character follows" (or indeed an extremely special one: "treat the sequence "\'a" as "á").

Recommendations: The description of any machine readable source should contain a list of all strings of characters, which are considered to form precisely one character logically. It should describe which of the above conventions have been used. When designing such systems, it should be taken in mind, that most transfer problems are made much

easier, if characters, which are not used elsewhere in the text, are used to announce that the following character is to be handled differently. Any convention of the form "every character following a square bracket shall have a dieresis ("[a. —> ¨a")" is much easier handled than one of the form "if a square bracket is followed immediately by a, o, u, A, O or U, the vowel shall be bearing a dieresis; else treat the square bracket as an independent character".

1.4 Do the data contain any white space characters (space, tabs, form feeds) which shall be ignored (conditionally or unconditionally)?

Many software systems, particularly from commercial environments, tend to add spaces or similar characters to the end of fields that are thereby artificially brought up to some predetermined length.

Recommendation: Any description of a machine readable data set should contain a note, at which points space characters may be discarded and at which not.

1.5 Do the data contain any symbols valid for printing, which shall be ignored and/or passed along for further processing during some later stages?

Data contain hints for printing basically in one of two forms: (a) either as a symbol indicating, that some alternative print mode becomes effective at the character following the symbol (size of print, font selected and so on), cancelling all earlier print mode commands which are concerned (a command dealing with the font selected, will certainly cancel out the last font selecting command, not necessarily, however, the command selecting the size to be used). The other alternative are (b) pairs of symbols, which state where a given print mode starts to apply and where it ends. Print control characters appear very often in the form of non-printable characters, which very often are not visible in the text if looked at with the help of some editor. (Such non printable characters are often known as "control sequences".)

Recommendation: Any set of machine readable source material should contain in its description, a list of all symbols announcing a change of the current print mode. If there exist more general definitions - "e.g. everything starting with a backslash is a printing command, ending by the first space or curly bracket ("{") after that backslash" - they should be included into the documentation.

Irrespective of such a more general definition, the list should contain for every print command the following information:

- *The character sequence that makes up the printing command.*
- *A short description what printing properties are starting with this symbol.*
- *The symbol that explicitly cancels the printing mode entered via this command.*
- *A list of all other symbols, which cancel implicitly the print mode entered by the symbol in question.*
- *A short statement, if this symbol (and the corresponding closing symbol) shall be dropped from the data, when they are going to be used for other purposes than printing, or if it shall be kept for other purposes as well.*

2 Special sorts of text

With the exception of the trivial case of a historical source entered into the computer in some Fixed Format, all historical sources, which shall be handled by DBMS-oriented software consist of a number of fields, containing texts of various meaning. These fields are separated by special characters, which may usually not be used within the fields. Exceptions can be provided for in three ways: either (a) a pair of symbols is defined, between which *all* characters are considered to be part of the textual characters or (b) one character is defined, after which every other character, including this announcing character, is considered to be part of the text or (c) a facility exists to redefine the set of field separating characters dynamically from within the input data.

Recommendation: Every description of a machine readable source should contain a list of all characters, which are used for the purpose of held separation. A list of the textual characters can be added, will usually be redundant, however, as the set of the allowed textual characters are simply those characters, which appear in the set of all defined characters (s. above 1.2) less the characters defined as field separators.

The description has furthermore to contain a description of all conventions used to escape temporarily from the definition of a held separator. If a facility for dynamic redefinition of the field separating characters exists, it should be described completely, that is, include a description of all the commands used to initiate a redefinition.

It is furthermore recommended to restrict oneself to the convention where each field separator used as textual character is preceded by an escape character. While the other two ways to handle this problem are much more elegant, the are available only in relatively few programming systems and can create cumbersome problems, if the system employing them has defaults for re-installing the original field separating characters by implication. Such is usually the case, when a major new portion of the data structure is being recognized. This behaviour of the software system for which the historical source has been prepared has, of course, to be part of the description of the machine readable material.

Independent of the kind of text stored within a given field, furthermore, all of them can be concerned by the following considerations, which are, therefore, not included into the recommendations for individual types of text, except in cases, where they have to be handled differently from the standard procedure.

In some systems a subset of the fields possible can be subdivided into multiple entries, that is, entries within the field, which are separated by a special character and can be considered as logically equivalent values of that field. Most systems provide, furthermore, the possibility to flag a field, that is, to add some characters treated specially by the system, noting, that its content is "questionable", "remarkable" or something similar. Such flags appear usually either as special flag signs - e.g. "?" or "!* - which are striped from the text before it is processed further, or as a special clause containing a validity description for the field, being either at the start or at the end of the field and separated from it by a special reserved character. On top of that some systems may add information to the field, by which it is defined, if the field belongs to a given subset of discrete "views" of the data base. Such views have been introduced for reasons of data security into DBMS software, can be used within historical contexts, however, to distinguish, e.g., between different layers of a source having been created over some time by successive contributors.

Recommendations: The general description of the machine readable source should for every field tell:

- *What the maximal length of the field is.*
- *How many logically equivalent entries it can contain.*
- *How long each individual entry may be.*
- *By which special character individual entries are separated.*
- *Which characters, if any, are system flags for that field and which semantic meaning the carry.*
- *Which characters, if any, separate a view and/or validity clause in the sense given above from the main body of the held.*
- *Can such clauses appear only at the beginning, only at the end or in both positions of the held?*
- *For both kinds of clauses, we recommend the following format: "clause-number-or-name separating-symbol clause-number-or-name separating-symbol clause-number-or-name ...". Clause numbers we understand to be natural numbers, which evaluate a property of the field along some numerical scale; clause names are supposed to be abbreviations taken from a finite list of such (e.g. abbreviations for scribe names). The description of a machine readable source making use of such constructs should contain the following items of information:*
  - *The character separating individual clauses.*
  - *For clause numbers: the maximal and minimal numeric value the can take.*
  - *For clause names: a complete list of all clause names used, together with an explanation of the meaning they are supposed to carry.*

2.1 Plain Text

We consider "plain text" to be the kind of text used to transmit natural language as appearing e.g. in charters, chronicles and similar sources. This kind of text is usually entered just as it appears in the printed edition. Three exceptions to that rule are made (relatively) frequently: (a) Sometimes editorial comments (e.g. the text of an critical apparatus) is entered within the main text. Such insertions are usually indicated by a pair of symbols, which define, where the editorial comment starts and where it ends, (b) Different qualities of portions of the text (falsified, emended, dubious) are indicated either by a system of brackets or by the insertion of printing commands, changing the appearance of the portion of text in question, when it appears in a printed edition, (c) In a few cases a text is prepared as a sequence of tokens of the form '"word-form separating-symbol lemma".

Recommendations: Plain text should be described by at least three lists of characters, which may be augmented by additional lists of symbols. These are:

- *A list of all "alphabetic" characters. An alphabetic character is a character, that is used to form a word. In this case diacritics are to be considered alphabetic characters.*
- *A list of all "word separators". A word separator is any character, that indicates the end of a word. This list should also include the characters out of which escape sequences indicating the start/end of the special sections of text discussed above are made of.*
- *A list of all "sentence separators". A sentence separator is any character, that indicates the end of a portion of text, that consists of one or more words and is a meaningful analytical entity which can occur more than once in the held, that is holding the text.*

- *A list of all symbols signifying the start/end of an editorial comment. This list should indicate, if editorial comments may be nested, and, if so, if any symbols indicating the start/end of one of the comments imply the end of another one.*
- *The character (if any) separating the word form in the text from its lemma, together with a statement, if the word form or the lemma comes first, plus a rule how cases shall be handled where only one part of the token "word-form separating-symbol lemma" occurs. In most systems known to the author such a situation implies, that the word form is identical with its lemma: this may not be the case in a particular application, however.*

2.2 Terminology

Terminological entries are distinguished from plain text in our definition by consisting of one ore more keywords or keyphrases, separated by a separating character. While editorial comments within keyword fields are not unheard of, the are unusual. Terminological entries in our definition have, however, two additional properties.

They are often intrinsically structured, that is, a keyword or keyphrase is often made up of a set of terms, which are in a hierarchical relationship between themselves and separated by a special character as e.g. in "agriculture/education of farmers/Saxony". Such structures assume, that software handling the terminological data is able to access this keyword equivalently as "agriculture", "education of farmers", "Saxony" and by all possible combinations of the constituent terms as well as by the chain of all three of them.

Additionally such terminological entries are often structured extrinsically, that is, the are related to a "terminological thesaurus", "thesaurus", "semantic network" or the like.

Recommendations: Any fields of data being terminologically in the sense of the description above should be described by the following information:

- What *is the maximal length of a keyword or keyphrase?*
- *How many keywords may maximally appear within one field?*
- *If more than one: by which character are the separated?*
- *How many terms may maximally be combined into a keyword or keyphrase?*
- *If such a combination is possible: by which character are the individual terms being separated?*
- *If any thesaurus describes the relationships between the keywords or keyphrases, it is recommended, to describe it as an independent historical data base. Most DBMS oriented software known to the author has distinct modules for loading such thesaurus structures. The description should - besides the other information contained in this document - clarify the following point: is the thesaurus distributed from the source system complete, or is the destination system expected to expand the directives given to it? (In the case of an relationship "x" which is valid between the keywords "A" and "B" the fact that "x" is an reciprocal relation will prompt many systems, as soon as the statement "A is related by x to B" is encountered, to create automatically the statement that "B is related by x to A". Some systems will react annoyed, if after this the fact that "B is related by x to A" is explicitly mentioned. A few will, however, expect this second statement to be submitted.*

  The minimal description of any such thesaurus should contain for each possible relationship that can exist in it, the following information:

- *Is the relationship reciprocal (follows "A related by* ***x*** *to* **B"** *from* **"B** *related by* ***x*** *to* **A "J?**
- **Is** *there a named inversion (follows "A related by x to B" from "B related by y to A")?*
- *Are there implied relationships for two terms, which both are related by the same relationship to another one (follows "A related by q to* C" *from "A related by x to B" and "C related by x to B")?*

2.3 Names

Names are entered either by assigning different constituent parts of the name to different fields (surname, middle name, Christian name, patronymicon etc.) or by assigning all that information to one field, which is structured itself according to some formal rule, e.g.: "The field starts with the surname given in capital letters (or substituted by a hyphen if missing), followed by a comma and the Christian names. If it is not sure, which of the later has been the first Christian name and which the second, they are separated by a semicolon. The maiden name, if known, is enclosed within a pair of brackets immediately after the surname, also in capital letters only. A nickname is suffixed to the name by the character ' = \ " (An entry according to this example could, e.g., be: "MAIER(HUOBER), Katharina = Huoberkathl".) In some cases, furthermore, some constituents of a name are stored as a sequence of lemmata.

Recommendations: In most cases assigning the constituent parts of a name to individual fields will reduce the difficulties when transferring the data to another software system considerably. When this logic is followed, the fields should simply be described, by giving the following information:

- *Does the field contain the name in the spelling of the source or in a normalized form? In the later case some software environments may profit, if a list of all the names that can possibly appear, is submitted as part of the description.*
- *Has any character been used to indicate, that a set of names (usually Christian names) can appear in any sequence to denote the individual? Which character has this been?*
- *If for purposes of sorting or comparison of names any (semi)automated solution has been applied to the data, the machine readable source will be considerably more useful, if this method is described in the documentation of the machine readable source. Do not, however, present this description only as a short program in the command language of the source system or some Higher Programming Language. While a copy of such a program may be helpful, it will usually be much easier to rewrite such a program from a good and thorough verbal description of its logic than by learning the command language of the source system first.*
- *If the names appear as a chain of lemmata, indicate:*
  - *by which character the constituent lemmata are separated from each other and*
  - *if within the string of lemmata some ordering has taken place (e.g., if the alphabetically smaller lemma appears always in front of the alphabetically larger one, independently of the order the parts of the name have in the unlemmatized name).*

If the names have been entered into a field with some internal structure provide additionally the following information:

- Give *a precise verbal description of the structure into which the constituents of the name have been entered. This is in most cases incomparably more helpful, than some system specific definition, e.g. the definition of a mask, by which the data have been entered.*
- *Explain for every constituent of the name in detail, what action has been taken, if this constituent has been missing in the original.*

2.4 Flags

Flags we call fields containing short abbreviations only. Three solutions have frequently been used: (a) Fields which can contain only one character, indicate by its absence or presence already the fact to be documented, (b) Fields which contain single letter abbreviations, which are combined irrespective of order to describe a set of possible values of a number of variables (CSM for Catholic Single Male, FPW for a Female Protestant Widowed person and the like), (c) Similar constructions have sometimes been use with abbreviations of varying length, being separated in combinations from each other by some special character.

Recommendations: The following information should be provided:

- *Describe, which character is expected to appear in this field to indicate the presence of a given property. Describe further, if the appearance of another character may indicate an error.*
- *Describe:*
  - *the set of all characters allowed to appear in this held,*
  - *describe within which subsets the characters are mutually exclusive. (In our example, one can not easily be at the same time M(ale) and F(emale).)*
- *Provide for the short abbreviations the same kind of information as given for b), plus a definition of the character separating the individual abbreviations.*

2.5 Auxiliary Attributes

Auxiliary attributes we call all entries, which are added to a textual field, without being text proper and without being a special flag character provided by the respective software system. Such additional entries, sometimes also known as "user-defined flags" do not form separate fields but are added immediately to another field, e.g. "Nicholas+" to signify a person already dead at the time when the source has been written or "John&" to signify a person which is a citizen of the local town. Few software systems provide special facilities for the handling of this kind of coding. Most of them prohibit it implicitly for the more formalized kinds of text which we shall discuss below.

Recommendations: Avoid this kind of input convention if possible. An additional flag held next to the name will produce only very slight overhead and increase the clarity and transportability of your input conventions considerably. If you think you can not avoid such a mechanism: use such auxiliary attributes only as suffixes, never as prefixes. For purposes of data transfer include a list of all characters that can be suffixed to a given field of information.

2.6 Calendar Dates

For the coding of calendar dates, which describe individual days within some temporal scaling with a well defined zero point, exist a wide variety of proposals, conventions and

practical solutions. We differentiate between the following ways of entering this kind of information:

2.6.1 Abstract time scales.

The most obvious solution consists in entering the dates as days expressed by an abstract time scale, like the Christian Era, the Islamite Era or the Revolutionary Era of the French Revolution. Such calendar dates are usually given as a string formed according to some abstract expression like "First comes a digit representing the day, than - after one or more spaces - a string of alphabetic characters, of which the first three represent an abbreviation of the month, followed - after one or more spaces - by the year, given by Arabic numerals".

Recommendations: Obviously any data entered like that have to be described by the precise format of the calendar dates. We have made the experience, that such "obvious" systems lend themselves quite easily to less than precise description, however. Points which can create quite some trouble, are particularly:

- *Have abbreviations for months to be given exactly or does some truncation take place? (Has J UN be noted as such, or would JUNE be legal?)*
- *Does case conversion apply automatically to such helds?*
- *Has the source system performed checking for precise dates?* **Warning: Target systems have been known to produce disastrous results, when they did assume, that all the calendar information supplied to them was free of errors, while the source system relied on built in checks to discover and eliminate erroneous information automatically.**
- *How are missing elements handled? This should be described very carefully. The three most frequent conventions for missing elements of calendar dates (e.g. unknown day, when month and year are known) seem to be: (a) substitution of a NULL element ("NO JUNE 1745", "? JUNE 1745", "0.0.1745"), (b) assumption of a NULL element, when a component is missing ("JUNE 1745" is equivalent to "NO JUNE 1745") and (c) reusage of missing components from the last (or next) complete calendar date entered ("12.,14. and 18.6.1745" being handled equivalently to "12.6.1745, 14.6.1745 and 18.6.1745"). The description of the input conventions should also try to describe the internal logic of the handling of calendar dates, if full replicability of a study is being desired. Some software systems will e.g. replace an unknown day of a month by the first day of that month, others by its fifteenth for purposes of some calculations.*
- *Does the treatment of dates change in some way with certain triggering dates, as, e.g., the date of the local introduction of the Gregorian Calendar? Are there possibilities for flagging exceptions from this assumption? Can the basic assumption be changed dynamically, either in relation to some other information of the data, or only by specifying, that from the given point of the data onwards, the default has to change?*
- *Is more than one timescale used side by side? (E.g. the Islamite Era side by side with the Christian one.) It is very strongly recommended, that in such cases a convention is used, which makes it possible to decide to which of the timescales a given calendar date belongs by simply looking at it. (e.g. by writing Christian dates like 12.6.1354, Islamite ones like 7 Ramadan 1245).*

- *When Roman numerals are legal for some component, it should be specified, if unorthodox* ways of writing *them are legal. (Is Villi the same as IX?)*

2.6.2 Relative Time Scales

Relative time scales, like the years of the reign of a given ruler ("12.5. Eliz I 7" indicating the 12 of May of the 7th year of the reign of Elizabeth I) are quite frequently used in the data processing of historical sources in countries where clearly established lists of rulers of a well defined political or spiritual realm exist.

Recommendations: As few existing software systems will be able to interpret automatically such lists from other countries, we recommend to split the description of such a system into two parts:

- *A description of the syntactical construction of the system used. This description should be checked against all the points we raised for 2.6.1, as quite frequently such a notation can be understood as a special case of the preceding one, with a particularly complicated way of specifying the year.*
- *A table specifying all abbreviations of rulers used. It should be specified, if the "first year of the reign" is counted from a given day or always from the first day of the calendar year in which the ascension took place. For checking purposes with sensitive systems, it will be reasonable, to include a definition of the duration of the reign assumed by the historian entering the data.*

The abbreviations used for rulers should be redundant, to avoid confusion of very similar names by systems truncating them.

Very great care should be taken, if any assumptions have been built into the source system, about the differentiation between two rulers with the same name (like: "the geographical expression mentioned in field x can be used to differentiate if the king of x or the grand duke of y is implied"). We recommend to differentiate between such names by different abbreviations.

2.6.3 Cyclical Time Scales.

By this we understand any system to specify a date by an expression involving a calendar of recurring events of clerical or secular importance. ("Sunday after Michaelmas.") Currently few software systems exist, which can handle calendar dates of this kind efficiently. The increasing importance of the possibility to make large corpora of diplomatic editions easily machine readable, will, however, in all probability increase the importance of features like this dramatically in the near future. We propose to use some amount of preediting for such data, as we think, that else the ambiguity of most of the events could be to dangerous to handle safely. A useful syntax to describe such calendar dates should follow the lines of "A phrase defining a day or a period of time, followed by a formalized way of describing if the period or day has been before or after the key event and, finally, a phrase describing a regularly reoccuring event, followed by a delimiter easily recognizable and after this the year." The second delimiter should be easy to recognize, because a calendar date like this will quite often be combined with components of a date using a relative time scale. (See 2.6.2 above.) Examples for such entries would be: "Fortnight past Michaelmas @ 1367" (for "a fortnight after Michaelmas has been celebrated in the year 1367"), "At Michaelmas @ 1367" and so on.

Recommendations: It is of utmost importance, that in any such convention it is absolutely clear, which list of reoccuring events is being referenced (e.g. which of the church calendars of several different dioceses). For all practical purposes, **a** *system which is sensitive for the context of the different fields, will be superior to one that is not, i.e., a solution, where by the content of some field of the input data it is clear, how the calendar dates of this type shall be understood. For reasons of portability we recommend, however, to specify a default calendar to be assumed and a flag in the formal definition of the calendar date specifying, if it is to be interpreted differentially. The recommendation under 2.6.2 to specify the formal definition of the date independently of the tables of phrases relevant for the specification of the different dates is* very *important here too.*

In all cases, where a less formal definition of such a calendar date is used, than the one mentioned above, a description of the algorithm used to convert the expressions into calendar dates of a more traditional outlook is necessary. As in other cases such a description should be given in natural (though formal) language, **not** *in the favorite programming language of the local EDP setup.*

More general: in all such cases all assumptions about the beginning of the year should be explicitly mentioned.

Irrespective of the kind of temporal scale used, the following three items of information should always be included with data to be transferred:

- *How are imprecise calendar dates being handled? This should not only discuss the formal description of the notational convention ("If only the terminus post quern of a date is known, it is suffixed by an hyphen ..."), but also the implications of handling imprecise v. precise calendar dates for the source system. (Is, e.g., an interval internally replaced by its median?)*
- *Are any fuzzifiers possible? ("Circa", "Approx") Here also not only the notational convention, but also its implications for the source system should be documented. (E.g: "Circa is described as an interval of n days around the calendar date following the fuzzifier")*
- *If the results produced by the source system shall be reproducable by the target system, it is important to specify clearly, if the source system assumes its data to form (a) a collection of data without any explicit temporal order, (b) a set of information ordered along an absolute timescale or (c) a collection of data, each of which can be attributed to a given point of time, these points of time not being strictly ordered, however. Please be aware, that a software system may not know – and care less - that its data are ordered in time though they are.*

2.7 Points of Time

Texts of this kind describe temporal events which can be determined more specifically than just for one particular day. As a rule, all that applies to "systems of measurement" (2.10/2.11 below) also applies to this kind of information.

Recommendations: To avoid unnecessary complications, we recommend to restrict oneself to one of the following conventions:

- *Give Time on the 24 hour time scale, separate hours, minutes and seconds by one separating symbol. For transfer purposes, this symbol should be documented and mentioned, if the timing information has been checked for impossible values.*

- *Give time on the 2 times 12 hour scale, separate hours, minutes and seconds by one separating symbol and suffix with an abbreviation like "am" and "pm". Specify for transfer: the separating symbol, the legal abbreviations, if there has to be a space (or a number of spaces) before the final abbreviation, which half of the day an item without the abbreviation is assumed to belong to and if an abbreviation longer than the minimum differentiating characters is to be truncated or treated as an error.*
- *Give time by a sequence of up to three items, separated by at least one space from each other, suffixing hours by "h", minutes by "m" and seconds by "s".* **Warning: A convention asking for minutes to be suffixed by ' or seconds by ' * is not to be recommended as conflicts with a language oriented use of these signs can easily occur.**
- *Use any of the above plus a list of additional keywords, which can be used as shorthand for a more precise expression ("noon"). Specify for each such expression, if it shall be interpreted as a point of time equivalent to the more precise form ("noon" being equivalent to "12:00") or as a somewhat less precise interval ("noon" being equivalent to "between 11:30 and 13:30").*

Treat in any case imprecise information, missing elements and fuzzihers according to the same rule as applied to calendar dates as described in 2.6 above.

2.8 Age and other Length of Time information

As a rule, all that applies to systems of measurement (2.10/2.11, below) also applies to this kind of data. We differentiate between these two special categories of text, mainly because all period of time related information is related not only to its own kind of notation, but also to the inherent logic of the treatment of calendar dates in the source and/or the target system of a data transfer. Length of time information is internally usually expressed as the number of days (in a few cases: hours or seconds) expressed by the specified text. To make temporal calculations sensible, it is expected, that calendar dates, before being used, are similarly converted into the number of days (hours, seconds) gone since some basic date (in historical research usually a fictitious 1st of January of the year 0 or 1, in commercial software either a date of the last decade or the 1st of January 1901 or the 15th of October 1582).

Recommendations: For any transfer of sources containing this kind of special text, two kinds of information have to be supplied.

- *The user has to be informed, what is the unit, into which all length of time information is converted internally. Please note, that we did not specify this when discussing calendar dates: indeed this knowledge is not necessary to transfer that kind of text as such and becomes only relevant if it shall be used together with length of time information.*
- *The user has to know, which notation has been used to enter length of time information initially. While all conventions discussed in 2.10/2.11 are applicable, it is strongly recommended, that a system is chosen, which describes a length of time as a string of items being separated from each other by one or more spaces and consisting of a number, immediately followed by a short abbreviation. The hrst letters of the various names of units of length of time in the national languages seem to be most appropriate, so "7y 4m 2w" would be a good transcription for 7 years, 4 months and*

2 weeks. When data are being described for transfer, it should be made explicitly, if these abbreviations have to be given as specified or longer ones will be truncated automatically.

Treat in any case imprecise information, missing elements and fuzzifiers according to the same rule as applied to calendar dates as described in 2.6 above.

2.9 Sequencing Information

In many historical sources, one wishes to use for identification purposes not simply sequential numbers, but structured numbers, which in the ideal case represent the labelling scheme applied by the archive the source material has been taken from. Such structured numbers consist of a series of subfields, which are either separated by a special character ("1-123-14"), by the different sets of characters they are taken from ("1123k") or by a whole range of separating symbols ("1/123-14,k").

Recommendations: To transfer data, for each system of sequencing information one should describe:

- *Which of these conventions has been used.*
- *Which characters separate the various helds. Please note, that, while most similar to archive labels, a convention using a whole series of separating characters runs most easily into conflict with systematically created systems of hierarchies of more global separating characters.*
- *For each held it is should be mentioned:*
  - *If the held contains true sequencing information, i.e. a systematically permuted combination of letters and/or digits, which in the case of "I-Berghausen-14-3" is only the case in the fields 1,3 and 4.*
  - *What kind of permutation is valid for this held (Arabic or Roman Numeral, single letter with duplication after z (= a, b, ..., z, aa, bb, ...), fixed length prefix with permuted suffix (CARA01, CARA02, ..., CARB01, ..., DIGA01, DIGA02, ...J etc.*
  - What *the maximum value of the field is.*
  - *What the minimum value of the field is.*
  - *What the maximum length of the field is.*
  - *If the field is left bound or right bound and if appropriate padding characters have already been inserted.*
  - *If it can be assumed, that successive sequencing items have strictly ascending identifiers or if the different sequential numbers follow in arbitrary order.*

2.10 Prices and Values

This category of special texts is separated from other systems of measurement by an implicit coherence: in theory at least every way to describe the value of an item of merchandise, the value of a tax to be paid, the value of an obligation under feudal law or the value of payment received by an laborer can be expressed in terms of each other. Similar in this respect to the various classes of temporal information, which we already discussed, values have in practical research no such obvious scaling dimension as time has.

To overcome this situation, values are usually "standardized" on three levels, each of which can be the level at which the data are actually input into the machine: values can

have a common (a) denomination, (b) base and (c) current exchange value. The common denomination of a system of words expressing a value just explains, how many of the units of one are the equivalent of the other: 240 denarii are (almost) always one pound. The common base leaves the level of units immediately deducible from a source and relates a system of value defining terms to some external dimension: expressing currencies not as multiples of some basic unit, but as some amount of silver or a similarly "objective" standard. The common exchange value of two items finally relates seemingly inconsistent items to each other: if a number of coins and a certain amount of grain are equally sufficient to fulfill a tenurial obligation, the are functionally equivalent for this source.

As all of these levels interact with each other, should not be interchanged conceptually, however, in most historical studies using the computer, which claim any sophistication at all, means were provided, to preserve the original notation of the source and to compute the meaning of the original words as values on some of the underlying dimensions later. While therefore superficially the task of describing the treatment a given source has undergone seems to consist in an enumeration of the input conventions, we would like to emphasize that the mechanical convention for preserving the names of currencies and the like are meaningless, if the decisions about calculations on the three levels mentioned introductorily are not documented along with the input conventions in a more narrow sense.

In this narrow sense, the following seem to be the most important conventions for entering value-related textual items. Lets start with the problem of denomination, helping the computer, that is, to understand a collection of wildly varying currency notations as multiples of some common denominator.

- Usually numbers entered with a decimal point identify just plain numbers; numbers, that is, in the decimal system. Historical software has been known, however, to allow for a redefinition of this convention, letting e.g. "12.4" mean "12 Taler and 4 Silbergroschen".
- Numbers entered with two decimal points are a more obvious deviation from the decimal world, signifying usually one of the older three value currency systems ("12.4.2" being 12 Pounds, 4 Shillings and 2 Pence).
- In both the last mentioned conventions, in some systems a comma is used to denote a decimal fraction of one of the non-decimal constituents of number: "12.3,5." representing, e.g., 12 Taler and 3.5 Silbergroschen.
- A more mathematically oriented way to handle non-decimal currencies consists in prefixing or suffixing a number with a radix abbreviation: "12.5" standing in such a case for 12 1/2 Taler, "12.5T" (or in some systems "T12.5") for 12 Taler and 5 Silbergroschen.
- A more systematical use is made of prefixes and suffixes by simply noting down the term used for a given currency in the original source: say "12.5 Friesacher Pfennige" with a meaning that probably has not to be translated.
- In all but the most primitive systems we find finally precautions for the unchanged taking over of such expressions like "12 Pfund Pfennige nach dem Würzburger Fuß weniger des Wertes von 3 Solidi" which usually are formalized by replacing the arithmetical operator by its symbol, like in "12 Pfund Würzburger Pfennige - 3 Solidi".

Such systems do not neccesarily **provide the same precedence of operators as other software - so it should be documented before data are being transferred.**

- **Our last examples were not very typical: there are systems, where the basic definition runs "a number followed by a** string **made up of non-digits means (number) items of** the entity described by that string". More generally is the convention, however, that **a** currency term may not have embedded blanks (many systems allow **instead** some filler characters, as, e.g., in "Pfund_Friesacher.Pfennige"). And a still substantial number expect the currency abbreviation to follow immediately, without intervening blank, after the number specifying how many units of this abbreviated currency shall be considered. **Warning: These differences are far from being trivial - the last mentioned systems may e.g. understand a chain of words separated by spaces as a case, where each word between two spaces stands for just one unit of the currency abbreviated by that single word, producing completely different computational results than the first one discussed.**

Recommendations: To exchange data, we propose to specify according to the catalogue just presented, which solutions have been taken. In almost all cases it is very dangerous to redehne the meaning of a single decimal point, as their remains practically no possibility to recover out of such data the few cases, where a number intended to be decimal has accidentally been entered with a redehned decimal point. It is a much more wise decision to leave numbers with single decimal points for truly decimal data and write nondecimal ones which shall be written without abbreviation always with two decimal points - even if that means, that you have to write "2.." for two pounds. It is not usually recommendable, to write multiple combinations of "value and abbreviation" without intervening blanks. "H2sh6p" is more endangered by the accidental dropping of letters than "ll 2sh 6p". The use of currency names with embedded spaces is usually a rather bad idea.

It makes data much more useful, when the syntax for the construction of proper values and the lists of abbreviations used are kept strictly separate.

The solutions for notifying **a** varying **base** for **the item in** question (changing content of silver of **a** nominally constant coin, agio/disagio **in** later times) **can be** grouped into three categories:

- Very often there is no difference being made between denomination and base of a given currency. That is, a "Friesacher_Pfennig" and a '"Würzburger„Pfennig" are simply being considered as two different multiples of a fictitious unit of "1/n ounces silver".
- A more dynamic solution - within data sets, where all the entries are ordered according to some timescale - is it to redefine the meaning of the currency abbreviations as often as necessary during the preparation of the input data.
- A further solution consists in the introduction of a special character, that separates an abbreviation for a denomination from another one for a base for that currency: "5Pfennig# Würzburger_Fuß" would clearly be a solution to differentiate between a "Pfennig" as a denominational unit and a "WürzbürgerJFuß" for the base of the Pfennig in question.

Similar solutions are obviously possible, if additionally the base abbreviation changes its meaning over time, i.e., if we have to distinguish between a "Würzburger_Fufl" of 1450 and one of 1470 onwards.

Recommendations: Dynamic redefinitions of the abbreviations used are the most elegant way to handle the problem; the are available only within very few software systems, however, and almost a guarantee for trouble during the exchange of data. Keeping separate sets of abbreviations for denominational conversion and currency base conversion is usually very recommend able. Do not forget to document which separating characters are going to be used.

For the final problem - exchange value no extensively tested solutions seem to exist. The problem is serious only in such cases, where there is no clearcut distinction between monetary values and material values to be received by some institution; say in cases, where there is reason to believe, that in certain medieval administrative documents an amount of grain to be paid by a peasant to his lord is simply an older way of expressing a standard monetary fee. The best solution up to now seems to enter all the values into one field, using a number of different tables, rather than to get out from the data at an later stage different sets of "values received". A more sophisticated solution can be realized with software systems, which can evaluate concurrently a whole series of input conventions, using for calculations at a given point of time only those item, belonging to a subset of these conventions. E.g. "12pound 4_busheLwheat 25#eggs" can under some software be evaluated as "12pound" or "4_bushel_wheat" or "25#eggs", the remainder being simply ignored, or any combination of such terms. Such a solution will be most flexible for the casual introduction of items, where the exchange value becomes known more precisely during the study. The software to handle such a system is scarce, however; this solution will require a very thorough documentation of the details of the conventions when the data shall be transferred between different software systems. A good idea would be, to enter the constituent items in a fixed order, as this can usually be easily converted into a string of subfields to be treated differently, which can be handled by a larger number of software systems.

Recommendations: Due to few standard solutions being available here, our recommendations have necessarily to be rather vague. In this context it will usually make very much sense, however, if the tables describing the various aspects of the system of values are described as an independent data base (and are included in machine readable form as well) as all but the most simple software should have some mechanism, to administrate such tables as dictionaries of some sort.

Treat in any case imprecise information, missing elements and fuzzihers according to the same rule as applied to calendar dates as described in 2.6 above.

2.11 Systems of Measurement

Most of what applies to currency and similar values, applies also to all the other systems of measurement, like weight, surface- or dry measures. The problem of exchange value is usually not very severe in this case: there exist instances, however, where measures change there meaning over time or have a different meaning in relationship to different goods the measure is used for. An additional problem exists, whenever a field in the data contains expressions, which are formed out of different systems of measurement. An

example of this would be a field, where a statement about an amount of money is expected, we find in the historical source, however, a phrase like "the money from the sale of 1 Viertel 260 Ruten of land, sold at 2 Taler 3 Silbergroschen each Rute, less the amount needed to buy 40 Malter and 5 Bushel of wheat at 1 Taler 2 Silbergroschen 3 Heller per Bushel". Due to all principles of source oriented data processing one would like to transcribe such a phrase something like "(lViertel+260Ruten) * 2.3. - (40Malter+5Bushel) * 2.3". Such expressions can technically be processed without much difficulty, create potential problems however, when it is necessary to apply, e.g., different dynamically changing bases to the monetary values and the dry measures appearing in the expression.

Recommendations: The documentation of the data should contain for every system of measurement a specification, if it can appear in any one field together with another system or other systems of measurement and, if there exist any means to recognize formally out of the term of measurement, to which system it belongs. For systems of measurement, where the precise numerical meaning of the term is dependent on the material to be measured, it should be explained, in which related field the description of that material can be found.

Treat in any case imprecise information, missing elements and fuzzifiers according to the same rule as applied to calendar dates as described in 2.6 above.

2.12 Locations, Occupations and other Categories

There exist many instances, where certain information is entered into the computer as a chain of short words, shall be treated however, as a set of numbers. Typical examples for such cases are provided by any terminology, which has to be classified before being used in statistical calculations, such as geographical regions, occupational codes and the like. Three cases can be differentiated:

- Data, where a simple one to one relationship between a term and the numeric code that represents it, exist.
- Data, where either the same field or another one has to be inspected in the light of another table or decision rule to produce the code to be used. (Say in German, where every occupation containing "meister" would be an independent artisan, while the "Vormeister" would be a military person.)
- Data there is additionally a (potentially weighted) many-to-many relationship between terms and classifying categories. (A place belonging with a probability of 60 percent to Prussia, which could, however, with a probability of 40 percent be in another country.)

Recommendations: In all these cases the tables containing the relationships between terminology and classifications should accompany the data as an independently described data base. Decision rules should be described in a formal, but natural language, not in some programming language. In the first of the three cases above, the transmitted data need to contain only the terminology; in the later two cases it is very advisable to include into the transmitted data alongside the terminology also the numerical classifications applicable in each case, as only very few software systems right now are able to process this kind of reasoning sufficiently easily.

2.13 Absolute Spatial Information

Absolute spatial information deals with geographical or topological information in a way, which allows precise distances between any two points to be computed. Such information appears in a source in the form of place names, names of landmarks, political or

administrative units or streets and houses within a city. It can be considered as absolute spatial information, if these names can be located on some map. This kind of spatial information exists internally, therefore, either as a point within some coordinate system or as a polygon described by a vector of such points. The connection between source expression and internal representation is usually created by a table which shows the equivalences between names and coordinates. Simple names of places usually do not change coordinates over time. Spatial stability is indeed usually considered as a defining property of, say, a city. Political units, on the other hand, change their spatial coordinates considerably over time and can only be represented by a set of coordinate vectors, where the appropriate vector is chosen after inspecting a field defining which temporal location the source entry in question has.

Recommendations: The tables describing the relationship between terminology and coordinates should be described as an individual data base. All the information about the properties of this kind of text should be put into this auxiliary description, with the exception of the rule, by which it is determined, which of a number of vectors, describing the borders of a political unit within time becomes applicable. All descriptions should make it possible, to treat the information within the main data base as plain text, as only very few of the existing programs make tools for the processing of spatial information immediately available. The following information should be contained in the description of the data base describing the relationships between terminology and coordinates.

- *Which coordinate system has been used?*
- *Where is the origin of the coordinate system?*
- *Which map has been used to take the coordinates from?*
- *In the case of local area maps, which have been turned into coordinates with a local origin of the coordinates and without precautions for the disturbance by the projection of the map: is there a known relationship between the origin of the map and the absolute altitude and longitude of earths surface?*
- *How many coordinates describe the longest possible vector?*
- *Can a spatial object be described by more than one vector?*
- *Is there a difference in the coding of vectors for which the last pair of coordinates is supposed to be connected with the hrst one (e.g. a border) and where this is not the case (e.g. a river)?*
- *How are potentially present plotting properties coded (e.g. the color to be used, if the spatial object shall be included within a map)? As plotting software is still extremely far from standardized, we recommend, to include as such properties only the following:*
  - *How is the symbol to be plotted at a given coordinate pair coded? Such symbols should be documented by a description in natural language plus an example from the plotting routine using it. If the formal description of the symbol is available to the researcher preparing the data base for exchange, it will considerably help to include either the generating formula or an vector with the coordinates of the symbol.*
  - *How shall lines to be drawn from a given coordinate pair be done? (Solid, dotted etc.)*
  - *Which color shall a symbol produced at a given coordinate pair have?*

- *Which color shall the line connecting the points of a vector have?*
- *Shall more than one line be drawn parallely when the points of a vector are connected to each other?*
- *In the case of "closed" vectors: by which shading pattern shall the area be hlled?*
- *In the same case: by which color shall the area be filled?*
- *If this point or vector is plotted so as to produce a conflict with other points or vectors within the data base just described, which precedence shall it take? (E.g., if a plot is used to present the result of a query and the administrative units within the area in question come out to be shaded: shall single villages, towns and cities within these administrative units continue to be plotted or shall the be suppressed?*

2.14 Relative Spatial Information

Relative spatial information deals with the location of spatial units, which can not - or in any case not at the beginning of a study – be related to an absolute location on a map, but relative to each other. (E.g. "house number two is to the left of house number one".) Such data are usually used to reconstruct a larger spatial network, they have typically a very variable quality, consisting of a combination of the following components:

- Landmarks, which can be expressed as absolute spatial coordinates.
- Landmarks which have no absolute spatial coordinates, but are stable reference points ("St. Martin's Church").
- Identifying names of individual places ("Foorler's house").
- Absolute directions ("North" etc.).
- Relative directions to places of known directions ("Left from x, when seen from the South").
- Relative directions to places of unknown directions ("Left").
- Vague directions ("Beside").
- Notions of proximity ("Close by").

There exist a number of (mostly tentative) solutions to handle special cases of such combinations appearing within historical sources, no general system, however, to integrate all of them into a software system. The following recommendations deal therefore not so much with the question, how such data can be transferred, but how the can be prepared in a way to be most completely and most easily used with existing software, while still being reasonably sure to be useful for software currently being developed.

Recommendations: We recommend to prepare such data for processing in a way which ensures, that all the kind of information discussed above, which deals with one physical object mentioned in the source, is a series of helds clearly related to each other. Abstractly, and without any prejudice for the different ways of structuring the input data for a certain piece of software, as described in chapter 3 below, we propose that all information dealing with relative spatial relationships should be organized according to the scheme:

Description of Reference Object 1
  Description of Relation to Related Object 1
    Description of Related Object 1
  Description of Relation to Related Object 2
    Description of Related Object 2

Every reference object and every related object should be described by at least the following information:

- *The name assigned to that object in the source.*
- *Its absolute location if known.*
- *A unique identifying tag, given to the object by the researcher. This is necessary, because it cannot be assumed, that names of objects within historical sources remain constant over time. On the use of these tags see below. If it is clear beyond doubt, that two objects are identical, they may be assigned the same identifying tag. For reasons of economy of work it seems to be sensible, however, to do so only in the case of obvious landmarks.*
- *For landmarks, where it is clear beyond doubt, that at certain sides there can be no adjacent objects, a held should be reserved, which contains those directions, where no immediately adjacent object can be situated. Such directions should be given in the same way as discussed below for compass points. (Such a direction would in the case of a city be dehned by an open square in front of a church or the side of the city wall, in the case of a territorial map by something like a lake.)*
- *Additional information may be added as necessary in additional helds.*

The relationship between any two object should be described by an arbitrarily long combination of directional items, which are separated by a special character (that is described in the documentation of the data base). The following classes of directional items should be used; which abbreviations are used, shall be described in the documentation of the data base. The vocabulary used for this purpose should therefore be controlled, not completely equivalent to natural language.

- *Compass points should be given by the abbreviations generally in use in the language of the researcher defining the data base (N, NE, NNE) or constructed according to the following principle: "Every compass point is defined by a single letter abbreviation. Every letter following another letter means the direction in the middle between the direction defined that far and the basic direction indicated by that additional letter." (NNE in the generally accepted meaning would of course be NEN in the second usage.) It shall be indicated in the description of the data base which of these conventions is being used.*
- *All other classes of direction should be given by natural language, however in a spelling, which makes them clearly discernible from compass points. We recommend to use lowercase letters for that purpose.*
- *In the case of a directional item like "left" being made more precisely by information from which direction the object is looked at ("left if seen from the South"), the abbreviation for the compass point should follow after a special character immediately after the directional term (e.g.: left:S").*

Additional information may be available and should be transmitted and described as a separate data base. In many cases it will become clear during a project, that different identifying tags of places refer to the same entity (building, farmstead etc.). In such a case the original tags should remain unchanged in the data to be exchanged and an additional data set, identifying the equivalences should be prepared. Such equivalences come usually from two causes: places being merged or split. If information about the time of such events

is available, it is best to integrate it into a data base describing for each "first" tag (i.e. for the first occurrence of a member of a chain of equivalent tags, representing an entity) at which times it "changes" by being renamed, split or merged.

2.15 Relations between Entities within a Data Set

Relations between two arbitrary items of information within a data set, which are not taken care of by the means used for structuring the content of a source as described in chapter 3 below, can be described by special texts of two sorts:

- In some software systems every item of information - or a complex of items of information - has an implicit symbolic address: "line 2345.14.4" or "the second son of the first spouse of the 25th family".
- In some software systems, the user has the possibility to define an arbitrary tag for a point in the data and reference it for purposes of connection: "This is meadow M457" and "Relate meadow M457 to the possessions belonging to the person currently being processed".

Recommendations: It is usually very hard to exchange the meaning of implicit symbolic addresses between software systems. If at all, this can only work if the rules for the construction of such addresses, that are valid within the source system, are explained in the greatest possible detail.

For data being referenced by tags contained within them, the following information should be part of the description:

- *How long may the tags be?*
- *Are they case sensitive?*
- *Are forward references possible (i.e., can the program be asked to connect something to meadow M457 before it has been described within the data)?*
- *Can such references be made within the whole data base or are the valid only within a certain context?*

3 Logical structure

In the preceding chapter of this paper, we defined a historical data base as being divided by a system of special characters into a number of fields, which contain different sorts of special texts. In principle there are two methods to do so:

- One can extract from a historical source the information needed, as, e.g., the names, occupations, places of birth and so on of the persons mentioned in the source, order these fields by special characters and submit them to the computer. (We call this "structured data" within this chapter.)
- Or one can transcribe the text of the source completely, in the order in which the words appear there, marking the beginning and the end of each portion of the text (names etc. as above) that shall be accessed and treated differentially from the remainder of the text. (This solution we call "text with embedded flags" within this chapter.)

Recommendations: Both methods are equally valuable. One should recognize the following consequences of selecting one of them:

- *Structured data are usually more quickly entered if taken from a handwritten original. They are also more easily cleared from logical errors. It is, however, not possible to reconstruct out of them the full text of the original source, the are therefore only in rare*

3.1.4 To reduce the number of incomprehensible symbols made up of special characters, it can be very useful to introduce different levels of symbols:

- a very abstract one, defining e.g. the character "/" as "here starts a new field" and the character "=" as "here starts the content of this field".
- a more substantial set of symbols, to be written and recognized only between the two symbols just discussed, like "surname", "price" and so on.

This logic, known as "Data list Input", "Descriptor/Descriptum" or "Tag/Content" logic, uses something that can also be described as a *generic symbol.* Such a system is usually not described by a list of all the symbols - like surname, price - being used, but by the rule "everything that appears after a BOR or a slash and before an equals sign is a symbol".

3.1.5 A rather obscure, but by no means unknown variant to this solution is a rule like the following: "everything after an ampersand and before a white space character (space, tabulation characters and so on) is a symbol. It describes the content of the field immediately before the ampersand, which starts at the first white space character before the ampersand." (E.g. "... during the further proceeding John&Christianmame Miller^Surname was accused&role to have trespassed ...".)

3.1.6 All the conventions we discussed that far did assume, that with every new line some kind of new entity started, which had to be divided further into smaller fields. In cases, where either a larger number of very small entities shall be processed, entities which can not be adequately described on one line or entities which will be described by a very variant amount of information, we need additionally a convention, by which we recognize the start of a new entity. Two solutions exist:

- A symbol (e.g. "@1") or a generic symbol ("everything after a BOR that is limited by a dollar sign") indicates the beginning of a new entity.
- Additionally to the symbols above another one (e.g. "BOR /") signals the end of an entity.

3.1.7 A further solution, applied most of the time to texts with embedded flags is presented by the logic of *start/stop symbols.* In it every field is enclosed between a pair of corresponding symbols. These symbols can either be simple ("everything between '&&P' and 'P&&' deals with a person mentioned in the text") or generic ones ("everything starting with the symbol and ending with the next white space character is a start symbol. The corresponding end symbol starts with the characters after the of the start symbol and is limited by '&&\ Stop symbols are sometimes universal (every is a stop symbol for the last 'fc^ix' encountered before).

Recommendations: The documentation of a data base should clarify the usage of the convention(s) used for partitioning it at least by:

- *Specifying all simple symbols,*
- *specifying the production rules for all generic symbols and*
- *commenting upon the decisions taken in all the points where in the discussion above alternatives have been pointed out.*

Additionally the following points should be mentioned as frequently causing misunderstandings:

- *Fixed Format: A few fixed format facilities provide for a dynamic change of the lengths of individual helds from within the data. ("The hfth held, containing the total of debts, is 5 digits long. If its hrst character is a comma, however, the Geld continues until the next comma is encountered.")*
- *Free Field Input: Be sure to enumerate all separating symbols. In a number of free held input facilities certain characters, particularly white space ones, always imply the end of a held.*
- *Data List Input: Be sure to include a detailed discussion of the conditions under which a previous value is "forgotten". Some data list facilities initialize every held with UNKNOWN when a new entity is encountered, some do not.*
- *The suffix convention described under 3.5 is a very good way to make your data hard to transfer between software systems.*
- *Mixed Forms: Most non-trivial software supports more than one of the input conventions described above. A precise description of the conditions under which a change between the modes takes place should be included. In some system special symbols exist which switch, e.g., between free held and hxed format.*

3.2 Identifying the portions of a source collection

For almost all non statistical purposes it is useful, to make in the resulting printouts the origin of the used data visible. (Registers, concordances etc.) For this purpose it is necessary to identify the various portions of the data. Three ways exist:

- The software system provides and maintains its own numbering scheme. (Counting the lines or the occurrence of various entities.)
- The user supplies at irregular intervals an identification, which will be printed along with all information retrieved for display purposes from the data between this identification and the next one.
- The user supplies at irregular intervals parts of an identification. Out of such parts the identification is being constructed. (Identifying e.g. the parts of the Bible by book, chapter and verse.)

Recommendations: The numbering schemes supplied by software systems make usually some assumptions which even the developers of the software may not be aware of anymore. When the data are being transferred to another software system, a user supplied identihcation should be entered. Please specify the maximal length the identification can take and supply all the information recommended in paragraph 2.9 of this document.

Many software systems can address directly (and interactively) the parts of the data which are identified by a given identification number. Quite a few of them differentiate, however, between identifiers which are used for such a purpose and such, which are just printed for reference. Identifiers used for accessing the data are usually supposed to be unique within the data base; identifiers just being printed are usually not. It should explicitly be specified which are which.

3.3 Structuring a Data Base into Documents

We have so far discussed our data as a sequence of fields, which are separated from each other and loosely grouped into entities. Contrary to that, software systems usually divide a data base into distinct data objects according to one of three models:

- Most often the data base is supposed to consist of *documents.* A document is a portion of the data which in the broadest possible meaning is "similar" to all the other documents in the same data base and has a unique logical structure. Documents are typically entries in a source representing a list (an entry in a register of marriages, a family in a census list, the transactions of a particular day within an accounting book) or one item out of a collection of similar ones (a medieval charter, a letter, a report on a particular trial). Within this model of a *structured data base* we can distinguish two different approaches.
  - In some cases such a data base consists of a relatively large number of logically different fields - say hundred or in some cases a couple of thousand - within each document. The individual fields, or at least their majority, have a relatively precise meaning (date of birth, surname etc.) and therefore usually a rather limited length. (Indeed some commercial products of this type think it a praiseworthy feature to handle fields of as *much (!!!!) as 256 characters.)* This type of data base is usually called *factual data bank* ("Faktendatenbank" in German), *knowledge bank* or known by some other name emphasizing the precise nature of the information contained within.
  - Contrary to this model, we have another approach, where each document consists of a portion of unstructured text - like an abstract - which is accompanied by a relatively small number of fields which contain a system of keywords classifying the content of that abstract and describing it in a very general way. This kind of system is usually known as a *textual data bank, information bank, literature bank* or known by some other name emphasizing the possibility to administrate portions of unstructured texts.
- In contrast to this first model, we also have *text banks, interactive concordances* or *full text retrieval systems* which do not subdivide the data the hold into anything resembling a document, but treat their data as a very long chain of individual words, which may have a very explicit location within the text such as "Spoken by Hamlet in the fifth scene of the first act of 'Hamlet"', be described by a whole array of formal ("past participle of a verb") or semantical ("pejorative attribute") properties. (For the purposes of this part of this paper, that is for a catalogue of the decisions to be described, when a machine readable source is being documented, we will in the following paragraphs consider a data base of this type to be a data base which contains precisely one document.)

Before we discuss our recommendations, we should note, that this describes of cause an *Idealtyp:* in reality most of the commercially available systems follow rather closely one of the three models described. Software developed within the historical disciplines, however, is quite often pretty hybrid with regard to these definitions, their respective limitations usually being one of the reasons why an independent development had been started.

Recommendations: We strongly propose to describe the basic properties of a data base along the lines drawn here. Information scientists pay usually very much attention to the abstract "data mode/", *which has been used when the software has been implemented. We assume, however, that a historian is much more concerned about the practical possibilities and limitations the software being used has - and when we are exchanging data, we are in*

any case interested, how we can fit them into the capabilities of our own software, not in the abstract principles of the design of the source system.

We recommend that each description of a data base to be exchanged contains the following information: if some of it is not available, the principal limitations of the source system with regard to the respective questions should be quoted instead.

- *Which of the three types described above represents the data to be exchanged most closely ?*
- *How many documents do the data consist of?*
- *Are there explicit or implicit references between documents?*
- *Do the documents have roughly the same length?*
  - *If so: what is it approximately? "Length" when referred to in the remainder of this document should be quoted either in bytes, characters or lines (accompanied by an estimate of the mean length of a line). If none of this measures is available - e.g.* 'every *document consists of three screens of data' - a very detailed description of the measure used should be included.*
  - *If not: what is the typical, maximal and minimal length of a document?*
- *Do the documents usually consist of the same number of helds?*
  - *If so: how many are there?*
  - *If not: what is the typical, maximal and minimal number of helds present? Are there helds, which are missing in more than ten percent of the documents?*
- *Which symbol indicates the start of a new document?*
- **Is** *there a symbol defining the end of a document?*
- *If the data contain an explicit tag or number identifying the individual documents:*
  - *where can it be found?*
  - *If not: which method has been used by the source system to assign identification numbers to the individual documents?*
- *Is it legal for data to appear between the end and start symbol (if the first exists) of two successive documents? If so: are these data to be ignored as comments or shall they be considered:*
  - part *of a general pool of unstructured information administered separately by the data base system,*
  - *a comment upon the preceding document or*
  - *a comment upon the following document.*
- *Are there any symbols within a document describing part of the data to be ignored as a comment? What are they (and their respective end symbols)?*
- *Does the source system consider its data to be related to a temporal and/or spatial scale? (That is: is it assumed to be known for every item of information, which date and which place or region it is belonging to?) If so, how is a change in time/and or region indicated?*
  - *Is every document considered to belong to one time/space frame?*
  - *Which fields contain the information regarding this?*
  - *Which fields, if any, indicate a change in the temporal/spatial references within a document?*

- *Are there any properties of tables related to the data base (dealing with the meaning of currency abbreviations, place names or anything else, which has been recommended to be described as a separate data base in chapter 2 of this document) which can be changed during the processing of input data?*
  - *If so: which symbols indicate the beginning and the end of the information describing such a change?*
  - *If this changing information does not follow the principles of structuring described when the respective table has been dehned, describe all commands to modify or replace one of the entries of the types of tables concerned.*

3.4 Structuring Documents

We have that far described documents as collections of fields, which may or may not be related to different entities. Usually these entities (persons, pieces of land, individual transactions and so on) will be related to each other according to some abstract model, which defines e.g. that a person may have a father, while a piece of land must not; within the data additional conventions exist to describe which person has which father. Within these entities a number of fields, describing these entities, is listed.

There are basically three ways to describe the relationships between different entities within the data:

- Entities have implicit or explicit identifications and contain fields which describe to what other entities they are related in which way. (Person P438 contains a field noting that its father is person P204.)
- Entities indicate by the sequential order of their identifying symbols, how they are related to each other. (A "father" following in the data the information dealing with "person" before another "person" is encountered, is assumed to be the father of this first person.)
- When both the start and the end symbol of an entity are contained within the start symbol/end symbol sequence of another entity, the former "belongs" to the later entity. (When after the symbol indicating that the following is a description of "person" the symbol indicating the start of "father" appears, before the end symbol of "person" has been encountered, father is assumed to be persons father. The following proposals, how these conventions should be described, assumes that the reader has made himself familiar with the recommendations for the description of fields containing different sorts of text (chapter 2 of this document) and of symbols identifying fields and entities (Introduction to chapter 3 and paragraph 3.1 of this document).

Recommendations: The structure of a data base should be described in the following way:

3.4.1 *Types of information used.*

- Start *with a description of properties common to all helds (or the vast majority of all fields), such as structure of subhelds and similar.*
- *Describe, according to the list in chapter 2 of this document, the rules pertaining to each kind of "special text". If there are some special texts, which fall into the same category, but differ systematically in some detail - e.g. two kinds of helds, which can contain currency values, some of them, however, containing currency in one notation and some others in a different one - list them separately. Assign to*

each *description some mnemonic name, like "plain text", "currency 1", "currency 2", "amount", "name" etc.*

- *Give* **a** *complete list of all individual fields appearing in any entity of the data. If* **a field** *can appear within 2 or more entities, is handled always by precisely the same rules, however, list it only once. If it is handled by different rules, list the rules common to all occurrences together and list than the solutions individual for the various applicable contexts. For each held mention:*
  - *the symbolic name of the held,*
  - *the start symbol for the held,*
  - *the end symbol of the held, if applicable,*
  - *maximal length of the held, if known,*
  - *the mnemonic name of the kind of information contained in the held from the list dehned according to the previous paragraph,*
  - *if this held has to be present and*
  - *how often this held may be present.*

For example:

Name	Start	End	Type	
Surname	**1st Free Field field**	**space**	**Name**	**has to be present**
First Name	**2nd Free Field field**	**space**	**Name**	
Date of Birth	**3rd Free Field field**	**space**	**Time**	
Occupation	**•'OCC^•**	**'•/••**	**Text**	**can occur arbitrarily often**
Taxes Paid	**▪▪**TAX=**▪▪**	**•7"**	**Money**	

3.4.2 *Types of entities used.*

List all entities appearing within the data. Proceed as follows:

- *Describe hrst properties common to all entities (e.g. a generic rule for start/end symbols).*
- **Define** *than a list of all classes of entities available. (A class of entities consists of all entities, which have the same helds.) Assign to each such class a mnemonic name such as "Person" or Property". If a number of entities have almost the same helds, describe hrst what the have in common, than any systematic differences.*
- *Give hnally a list of all entities existing containing:*
  - *the symbolic name of the entity,*
  - *the start symbol for the entity,*
  - *the end symbol for the entity, if applicable,*
  - *the mnemonic type of the kind of entity from the list dehned according to the previous paragraph and*
  - *any helds within this entity, which are different, when the appear here than when the appear in other entities.*

For example:

Name	Start	End	Type	
Person	(P	P)	Person	
Father	(F	F)	Person	Sex may contain ''D'* (for dead)
Field	(FL	FL)	Property	
Meadow	(MD	MD)	Property	

3.4.3 *Relationships between helds and entities.*

Dehne either as a set of general rules or as rules applicable to the different classes of entities dehned above:

- *If the last held of an entity ends with an explicit end symbol or this is implied by the end symbol of the entity.*
- *If there are any properties of the entity describing the entity as a whole ("written by scribe xxx", "emended").*
- *If (in a mixed input convention):*
  - *hxed format helds have to be present or a symbol exists indicating the end of hxed format and the start of free held and/or data list fields.*
  - *if missing free held helds have to be indicated by dummy values (0, "unknown") or if they can be indicated by multiple occurrences of the separating symbol and*
  - *if at the end of a list of free held helds empty helds have to be indicated before a name list portion of the data starts.*

3.4.4 *Relationships between entities.*

Dehne for each entity:

- *To which other entities it* **may** *be connected by which means,*
- *to which other entities it* **must** *be connected,*
- *if to any entities it must be connected to, it may be connected to only a given number of times,*
- *if to any entities it may be connected to, it must be connected to at least* **a** *given number of times,*
- *if the connections to other entities must appear in a given order (e.g. all children of a person have to be mentioned before the list of the servants starts) and*
- *in the case of start symbol/end symbol conventions, where the relationships of entities are indicated by inclusion, if entities may overlap partially. This case is relatively rare: assume an input convention, by which two entities "person" and "property" have the start/end symbols "(PER PER)" and "(PRO PRO)" respectively. All helds are closed by the character "/" and indicated by generic symbols of the form "a string of letters hnished by a colon". Can in such a situation a piece of property, which belongs to a person, about which, however, also information is available, which is not related to this person, be described by the construction:*

(PER surname:Smith/first:Joseph (PRO type:meadow/value:23.4.4 PER) comment: is this the place mentioned in ABC Nr. 231/b? PRO)

The description will be considerably shorter, if the relationships are described systematically, i.e., if one starts by saying, that to every entity of class "person" there may, but need not, exist an entity "Migratory Act" which may appear an arbitrary number of

times, instead of defining this to be the case separately for every entity of class "person" in the data.

3.5 Relationships between Data bases

That far we have discussed software systems which handle monolithic blocks of information: what is known about a given person is in the data being available or it is not. Such a design assumes, that all information to be handled is within one physical unit which for purposes of the exchange of data is transmitted as one distinct file. There are software designs, however, which deviate from this basic strategy: in such a system information regarding persons would, e.g., be stored, administered and maintained within one data base, information about properties within another one. If one wants to know, which persons owned which properties, one adds to each entity describing a person a field which contains the identifications of the properties within the second data base. Such a design implies quite a few technical problems - e.g., when in the property data base any piece of property is deleted, it should be guaranteed, that all references to that piece of property are deleted automatically from the data base dealing with persons.

Such designs originate from three considerations: on the one hand there exist certain commercial systems (using the so called *relational model)* which internally administrate a data base precisely in this way, even if the user is not aware of this. If data from such a data base are transferred to another user, it can be practical to transmit them in a number of different files. The second consideration is more generally: in quite a number of research situations it may become sensible to collect certain types of information independently from each other - for example because the sources which contain that information reside in different archives. This can even mean that in cooperative projects two data bases are collected and prepared at two different universities. And there exist, indeed, designs which allow two data bases to reside on different computers, which call each other every time a user accessing one data base needs a piece of information connected to it but residing at another site. A third consideration, finally, is that in large projects it may become sensible to administrate the data available by a "data base of data bases". This could imply, e.g., that a user accesses the master data base, telling it that he or she wants to know something about "all persons living within a given area between 1710 and 1720". Such a query could first be processed by examining a master data base for data bases containing such information, forwarding the query afterwards to all the data bases that can possibly fulfill it.

These last considerations are more or less Utopian today, as all such systems are in experimental stages right now. As such designs should become very prominent, however, if machine readable historical sources become available for secondary analysis more readily, we would like to add a few comments how such a system of interacting data bases should be documented.

Recommendations: Describe the different data bases hrst of all without caring how they are linked to each other. The helds containing hints to further information being contained within another data base should be entered and described in the following fashion:

- *If the second data base, the hrst one is linked to, resides on the same host computer, the held should contain information about the place the information is to be found in the second data base structured and documented as in paragraph 2.15 of this*

document. This information should be prefixed by a symbolic name for the second data base, separated from the remainder of the field by a special character reserved for this purpose. Alternatively the documentation can contain an explanation, that all information contained within this held references the second data base. Designs which make the location of a reference deducible from the structure of the reference itself should be avoided to eliminate potential causes of confusion. The documentation should furthermore contain a list, which describes the symbolic names of all data bases referencing each other and describes their physical characteristics at the time of transfer. If the system of interdependent data bases is consisting of more than two or three data bases, this list should be transmitted and described as an additional data base.

- *If the second data base resides on another host computer, (e.g. if only part of a system of interrelated data bases is being transferred, the remainder being accessible only at the original installation,) a further special character should be reserved to separate an identihcation of the host system prehxed to the expression described in the previous paragraph. All that has been said in this paragraph about identihcations of multiple data bases applies mutatis mutandis also to the identihcations of different host systems.*

3.6 Simplifying the Input

All the more sophisticated systems used for the processing of machine readable source material contain some means for simplifying input. As far as such means are restricted to the usage of masks for on line input, the are irrelevant for the transmission of data, as in all systems known to the user the data passed from the mask oriented input routines onward to further processing contain the entered information in the form it would have had, would it have been entered without any abbreviations. There exist a whole array of possibilities, however, for entering data in an abbreviated form, which is expanded and interpreted only, if the input data are read by the software system used for analysis. These aids for input have to be understood therefore by any target system intended to analyze further data coming from a source system with such capabilities. Please note, that masking facilities which are *not an* **integrated part of the software system used for analysis pass all input simplifications forward to the analytical system without expanding them: in case of doubt, the data should be consulted before transmission to the target system to check if the contain any such simplifying construction.**

Recommendations: The description of data to be transferred should contain a statement describing the precise usage of any of the following input aids, if available within the source system.

- *Which conventions for the continuation of lines are supported? Three conventions are in general use. All of them assume, that a field is continued across an EOR. The following considerations are of no concern therefore, if a BOR is followed immediately by a symbol indicating or implying the end of the held being processed at the preceding EOR.*
    - **No** *continuation across lines is possible. The EOR is replaced by a single space (or white space character).*

- *Continuation is purely mechanical. Every EOR within an held is simply ignored. This option can be implemented with two options: (a) most often all white space characters between the last non-white space character and the EOR are ignored. If an EOR coincides with a space, that space has therefore to be written at the beginning of the continuation line, (b) Some systems assume an EOR precisely and only after a hxed number of characters. In such cases spaces at the end of a line will be taken over into the data in the held.*
- *Continuation is triggered by a continuation character. (The "typewriter rule": if the last non-whitespace character before an EOR is a reserved special character - usually "-" - it is dropped and the line is connected to the next one without intervening spaces.) If the last non-whitespace character is anything else, the EOR is replaced by a single space.*

While this last rule is usually the most intuitively appealing, it creates the problem, that the continuation indicator may be a character, which appears very frequently within some helds of data - say the "-" within helds containing numerical expressions using the minus sign. Therefore it should be distinguished between systems which

- *perform the line continuation services independently of the held being read and such*
- *which follow different line continual ion rules dependent on the held currently being read.*

- *Do the data contain any global symbols to be expanded? Some software systems allow means to specify symbols which can appear anywhere in the input data to be expanded into a more easily readable term - if within a data base a geographical term like "Austro-Hungarian Monarchy" occurs very frequently, it may be desirable to enter it as AHM. To improve the readability of the results, it may, however, be equally desirable that the software system expands this abbreviation to its full length whenever it brings it to the knowledge of the user. Please note, that it is of no relevance for transfer, if such an expansion takes place when the input is converted into some internal form or when it takes place only if the expression is to be output to the user in some way. There exist two ways to support such a service:*
  - *Sometimes the abbreviations are expanded whenever they occur, being surrounded by two characters which are not from the set of characters out of which expandable abbreviations may be constructed - usually the letters of the alphabet.*
  - *More often the are only expanded if the are prehxed (or suffixed) with a reserved character or symbol. ("SJAHM" being, e.g., expanded, "AHM" being not.) This system is to be preferred.*

  The description of a data set containing such constructions should contain:
  - *a list of all characters out of which abbreviations may be formed,*
  - *a list of the abbreviations actually being used and*
  - *the character qualifying a character string as abbreviation, if any exists. (Of course it should also be explained, if it is used as a prehx and/or a suffix to the abbreviation to be expanded.)*

- *Do the data contain any symbols to be expanded conditionally? Some software systems perform the expansion service described in the preceding paragraph differently for different helds of the data. In such cases all that has been said in the preceding paragraph applies. The documentation should in such cases contain separate lists for all sets of helds, where different lists of abbreviations apply.*
- *Do the data contain any helds which dehne expandable symbols? While in most systems offering the services described in the two preceding paragraphs, the expandable abbreviations have to be given at the beginning of the data set to be processed, in some systems new abbreviations can be introduced - or previously used ones redehned - within the input data. All symbols triggering such a service - and the helds where they may do so - have to be part of the documentation.*
- *Which assumptions are made regarding the repetition of previously entered helds? Most non-trivial software systems contain precautions for the handling of data which can be taken over from previously entered lines or entities. (So the user has, e.g. not to reenter the surname of a family for every member when transcribing a census list.) Such a repetition service can take two basic forms:*
  - *//' a held is not explicitly entered within an entity, it is taken over from the preceding entity. (If no surname for a "person" is given, it is assumed to have the same surname as the preceding person.) For most historical data, this is a very dangerous service. This system can considerably be improved, if the helds concerned are re-initialized at some regular interval. (If, e.g., the surname of the last "person" becomes forgotten, as soon as the next document starts.) As experience shows, that the users of such services tend to be not aware of their existence, it would not worsen the documentation of a data set to explain that the source system does not perform this service and empty helds must therefore not be replaced with the value by the target system.*
  - *If a held contains a specific symbol, defined either for the data set as a whole or for a set of fields specifically, the content of some previous field is repeated. Here we have to differentiate between three possible solutions:*
  - *If a field contains the symbol triggering the repetition service, the content of the held having the same sequence number within the preceding entity is being repeated, ignoring any differences between the two helds in two successive entities which assign different symbolic names to such helds.*
  - *If a held contains the symbol triggering the repetition service, the content of the last field having the same symbolic name is being repeated, irrespective of differences between the entities containing these helds.*
  - *If a held contains the symbol triggering the repetition service, the content of the last held with the same symbolic name within the last entity with the same symbolic name as the current one is being repeated.*
  - *In all these cases periodic re-initializations may or may not be performed analogously to the description given for the first model of field repetition.*
  - *In a few software systems whole entities (or sets of entities) may be repeated. This is usually done by modifying slightly the symbol identifying the start of the entity. Such solutions being very specific, a description of the logic behind this*

service in a formal but natural language no local favorite programming language - should be included.

3.7 Data Integrity

In theory it is unimportant for the transfer of data from a source system to a target system, if the conform strictly to the specifications contained in the description of the data or to common logic. Experience has indeed shown, that particularly data coming from software systems which are closely oriented towards programming languages - as compared to program systems -, are virtually never in complete accordance with the specifications accompanying them, as most software is not guaranteed to discover all violations of the input conventions being used. As the step-by-step discovery of such violations during the transfer of the data into the input format of the target system may be extremely frustrating and time consuming, it is recommended that every data set to be exchanged be accompanied by a description of the checks performed explicitly or implicitly by the source system.

4 Prototypes of Information Contained in Historical Sources

The following prototypes shall constitute a basic grammar of entities which can be used to make a machine readable source understandable for a computer. They are described as a set of entities, which are being constituted by fields describing a core of information relevant to them. The intention behind these prototypes is the following: to provide some initial work for a catalogue of the classes of entities appearing in machine readable sources similar to the catalogue of individual fields provided in chapter 2 of this document. As the individual entities fields can consist of very many fields, these descriptions are, however, much more preliminary than the ones given in the chapter just quoted. This catalogue is supposed to be a suitable starting point for an agreement upon the "most preferred entities" that should be supported by a software system tuned to the exchange of historical data. It is assumed, that the are handled as follows, if adopted for exchange: each prototype is accepted as a standardization proposal, consisting of a set of symbolic names, each describing a field containing information of one of the types defined in chapter 2. All tables refered to in this chapter are referenced by another set of symbolic names. To implement such a standard the following procedure would be necessary: for each data set to be transferred it is explained, how the source system recognizes the beginning and end of the fields contained in the prototype, referencing them by the symbolic names used there. For each type of special text, as defined in chapter 2, a description of the conventions of the source system is given. Finally a description of all the tables used is given in chapter 5, referencing them by their symbolic names.

The conversion of the data should proceed like this: a software system [1] translates the source data into the representation necessary for the target system, restricting itself to the fields defined within the prototypes. Parallely to this conversion, the tables necessary for the interpretation of the data are converted as well. As in this approach all the conversion steps are processed very modularily (making recognition of fields, representations of

[1] A definition of a prototype of such a system is currently prepared by the author for presentation of this years workshop on standardization and exchange of machine readable data to be hedl in Paris.

different fields and tables independent of each other) software of this type should make the development of routines very easy, which perform at least some of this tasks independently of the source being processed.

This is partially of course a virtue of a specific software solution, but considerably smoothed by the existence of prototypes, as with them it becomes possible to restrict the conversion of data at the beginning to a core of "most relevant information", ignoring additional fields being in use locally, and postponing the conversion of the remainder of the information within a given data set until some later stage. As this remainder should contain most of the particularities of a local software system this procedure should speed up initial data conversion very much indeed.

This is all the more the case, if software developed for historical research purposes supports the use of the information defined in such prototypes. This would mean, that as long as someone is using the kind of prototype proposed in this document, he or she does not only increase someone else's potential for using the data – a rather low incentive as experience shows - but reduce as well his or her costs for preparing the data. To make this possible, newly written software for historical research needs some mechanism for this purpose.

All the prototypes are defined by schemes which give for each field the symbolic name of the field, the class of special text this field contains, and the symbolic name of the table used to interpret the content of this field.

4.1 Persons

Name of field	Name of special text	Name of table
Surname	Name	Dialect
Maiden Name		
First Name	Name	Dialect
Second Name	Name	Dialect
Sex	Flag	Sex
Religion	Flag	Religion
Marital Status	Flag	Status
Age	Time interval	Time
Occupation	Terminology	
Date of Birth	Calendar Date	Calendar
Date of Marriage	Calendar Date	Calendar
Date of Death	Calendar Date	Calendar
Place of Birth	Spatial Reference	Geography
Place of Marriage	Spatial Reference	Geography
Place of Death	Spatial Reference	Geography

4.2 Immobile Properties

Name of field	Name of special text	Name of table
Quality or type	Terminology	
Value	Currency Value	Price
Size	Measurement	Area
Location	Spatial Reference	Geography

4.3 Mobile Properties

Name of field	Name of special text	Name of table
Designation	**Terminology-**	
No. of pieces	**Measurement**	**Number**
Value	**Currency Value**	**Price**
Amount	**Measurement**	**Dry**

4.4 Careers

If the attributes of any of the prototypes listed above change at intervals, that is, if a person gets another job, changes its place of living or if a piece of land changes its owner, it is recommended, to describe such events in the form of a career, that is, to describe at the beginning the initial state of the entity and add afterwards entities, which consist of a field that describes the date the attribute in question changes (type *calendar date,* table *calendar)* and as many fields as there are attributes potentially changing. The prototypes for this entities describing a change consist of the prototypes of the entities for which the change shall be described plus the additional field date *of change.*

5 Prototypes of Tabulated Information

5.1 Dialect

This table, if present, defines the properties of the dialect in which the names of the area the source comes from have been transmitted. While the standard form of defining this table is open to agreement, it should contain:

- Information on which letters tend to be interchanged.
- A list of prefixes, suffixes and infixes to be ignored.
- A list of prefixes, suffixes and infixes to be changed prior to processing the name.

5.2 Sex

This table informs about the codes for sex being used within the data set. These codes have to be single letter abbreviations. It consists usually of codes for the following semantic categories:

- Female.
- Male.
- Child.
- Person of unknown gender.
- Group of persons described by an entity of class "person''.

5.3 Religion

This table informs about the codes used for religion. It consists of single letter abbreviations, each identifying one religion or confession. It usually contains codes for a subset of the following semantic categories:

- Lutheran.
- Catholic.
- Calvinist.
- Zwinglian.
- Anglican Church.
- Unspecified Protestant.

- Orthodoxy.
- Armenian.
- Jew.
- Islamite.
- Other.

5.4 Status

This table informs about the codes used for marital status. It consists of one letter abbreviations, each identifying one marital status. It usually contains codes for the following semantic categories:

- Single.
- Married.
- Widowed.
- Divorced.
- Separated.
- Unknown marital status.

5.5 Time

This table informs about the abbreviations used for the different intervals of time. It usually contains abbreviations for a subset of the following semantic categories:

- Year.
- Month.
- Week.
- Fortnight.
- Day.
- Hour.
- Quarter of an hour.
- Minute.

Additionally it contains information about the way a number without any abbreviation shall be treated, i.e., which unit of time shall be used as default for this field.

5.6 Calendar

This table informs about:

- The abbreviations of months acceptable.
- The date of the introduction of the Gregorian Calendar, if it falls into or before the period covered by the data.
- The abbreviations acceptable within relative and/or cyclical chronologies plus the syntactic rules of such chronologies.

It is usually accompanied by an explanation in natural language, which calendar(s) may be used within the field.

5.7 Geography

This table contains absolute spatial coordinates for each geographical term appearing in the field, optionally enhanced by information how the geographical term shall be plotted into a map.

5.8 Price

This table contains information about:

- How many units each abbreviation of a currency in the field is to be considered to consist of.
- How numbers without any abbreviation shall be treated. (I.e., what is the default unit of the field.)
- How numbers with one decimal point shall be treated.
- How numbers with two decimal points shall be treated.

5.9 Area

This table contains:

- For each abbreviation being used its equivalent in square meters.
- For numbers without abbreviations the area they represent in square meters. (I.e., the default unit of the field.)
- For numbers with two decimal points:
  - How the shall be converted into decimal units.
  - How many square meters each of these units represents.

5.10 Dry

This table contains:

- For each abbreviation being used the number of liters it represents.
- For numbers without abbreviations the number of liters the amount to. (I.e., the default unit of the field.)

5.11 Number

This table contains for enumerative units ("dozen") the number of items they represent.

6 Prototypes for the Structures of Historical Sources

The prototypes defined before can be used as building blocks to create prototypes for more complex entities, e.g. whole sources. The prototypes presented in chapter 4 have been tentative ones, just to introduce the principle of prototyping. This is even much more so the case with the more complex prototypes to be described now. The reason for their introduction, however, is precisely the same as discussed in the introductory remarks of chapter 4. Please note, that these prototypes are intended to be something like the smallest common denominator: the author is aware, that in all projects he knows which are using this type of source material, more information is taken from the source in question.

These prototypes draw heavily from the experiences gained with the processing of various types of source material with CLIO in Göttingen since 1978.

6.1 Tax Lists

The prototype of a tax list is defined by a series of entities with the same starting symbol. These entities are of the prototype "person" enhanced by two fields:

Name of field	Name of special text	Name of table
Identification	Sequential Number	
Amount Paid	Currency Value	Price

6.2 Census Lists

The prototype of a census list is denned as follows:

Each cohabitational unit of the census is an individual document. It starts with an entity of the prototype "immobile property" enhanced by the following field:

Name of field	Name of special text	Name of table
Identification	Sequential Number	

This entity is immediately followed by three entities of the prototype "person", being defined as "Owner", "Head of Household" and "Wife". After these three entities follows an arbitrary mixture of other entities of the prototype "person" being defined as "Son", "Daughter", "Relative of the Head of Household", "Relative of the Wife", "Servant", "Co-inhabitant".

6.3 Parish Registers: Baptisms

Each document represents one baptism and is introduced by an entity with the following fields:

Name of field	Name of special text	Name of table
Identification	Sequential Number	
Date of Birth	Calendar Date	Calendar
Date of Baptism	Calendar Date	Calendar
Illegitimate?	Flag	
Stillbirth?	Flag	

This entity is followed by entities of the prototype "person": "Child" and " Mother", each of which has to appear once and only once, not more than one entity of the same prototype "Father" and an arbitrary number of entities of the same prototype "Godparent".

6.4 Parish Registers: Marriages

Each document represents one marriage and is introduced by an entity with the following fields:

Name of field	Name of special text	Name of table
Identification	Sequential Number	
Date of Marriage	Calendar Date	Calendar
Place of Marriage	Spatial Reference	Geography

This entity is followed by entities of the prototype "person": "Bride" and "Bridegroom", each of which has to appear once and only once and may be followed by one entity "Mother" and "Father". After this an arbitrary number of entities "Witness" may follow.

6.5 Parish Registers: Deaths

Each document represents one case of death and is introduced by an entity with the following fields:

Name of field	Name of special text	Name of table
Identification	Sequential Number	
Date of Death	Calendar Date	Calendar
Date of Burial	Calendar Date	Calendar
Place of Death	Spatial Reference	Geography

It is followed by entities of the prototype "person": exactly one "Deceased", optionally one "Mother" and "Father" each and an arbitrary number of "Spouse's" and "Witnesses".

6.6 Court Records

Every case tried forms one document. It is introduced by an entity with the following fields:

Name of field	Name of special text	Name of table
Identification	Sequential Number	
First session	Calendar Date	Calendar
Last session	Calendar Date	Calendar
Subject	Terminology	

This entity is followed by an arbitrary number of entities of prototype "person" which have the following additional field:

Name of field	Name of special text	Name of table
Role at Court	Flag	**see** below

This additional field contains single letter abbreviations, which usually form a subset of the following semantic categories:

- Judge.
- Public prosecutor.
- Lawyer of defendant.
- Other lawyer.
- Other court member.
- Juror.
- Defendant.
- Party
- Party 2.
- Party 3.
- Party 4.
- Witness of prosecution.
- Witness of defendant.
- Unspecified witness.

This is followed by an entity "summary" which consists of only one field:

Name of field	Name of special text	Name of table
Text	Plain text	

6.7 Testaments and Inventories

Every testament or inventory forms one document. It starts with an entity of the following form:

Name of field	Name of special text	Name of table
Type of document	Terminology	
Date of document	Calendar Date	Calendar
Place of document	Spatial Reference	Geography

The remainder of the document is made up of entities of the prototypes "person", "mobile property" and "immobile property". Each of these entities has additional fields:

For "person":

Name of field	Name of special text	Name of table
Identification	Relations between entities	
Related to	Relations between entities	
Inherits	Relations between entities	

For "immobile property" and "mobile property":

Name of field	Name of special text	Name of table
Identification	Relations between entities	

Immediately after the first entity follow one or more entities of the enhanced prototype "person" known as "testator". After each of them follows a list of arbitrary length of entities of the enhanced prototypes "immobile property" and "mobile property".

Now follows a list of entities of the enhanced prototype "person" known as "creditor" and "debtor". Beyond the enhancement noted already, these entities have additionally the field:

Name of field	Name of special text	Name of table
Amount owed	Currency value	Price

Finally follows a list of arbitrary length of entities of the enhanced prototype "person" known as "child", "relative", "other heir".

The additional fields are used as follows:

Entity	Field	Usage
Testator	Identification	Unique id for person
	Related to	No standard usage
	Inherits	Identification of property, credit or debt inherited from other testator
Mobile Property/ Immobile Property	Identification	Unique id for property
Creditor/Debtor	Identification	Unique id for person
	Related to	Identification of person to which/ by which ''amount owed'' is owed.
	Inherits	Identification of property, credit or debt inherited.

Child/Relative/ Other Heir	Identification	Unique id for person
	Related to	Identification of person to which this one is a child or relative.
	Inherits	Identification of property, credit or debt inherited.

6.8 Chronicles

To be maintainable by data base systems administering collections of documents, and to provide at the same time as complex a context as possible, a chronicle or similar source is divided into "documents" which are defined by some criterion, depending on the source in question: when applicable, the year covered by a number of paragraphs, a count of chapters, books or similar should be used. If no such structure exists, a page of the printed edition becomes a document. Every document starts with an entity of the form:

Name of field	Name of special text	Name of table
Identification	Sequential Number	

and consists of entities, which describe precisely one line of text in the printed edition. These entities have the form:

Name of field	Name of special text	Name of table
Line reference	Sequential Number	

If documents are formed according to years, chapters or similar, it is assumed, that the "line reference" reflects the page number on which the respective line appears.

6.9 Accounting Books

Every transaction recorded in an accounting book is considered to be a document. It is described by an entity which consists of the prototype "mobile property" enhanced by the following fields:

Name of field	Name of special text	Name of table
Identification	Sequential Number	
Date of transaction	Calendar Date	Calendar
Received/Expended	Flag	
Purpose	Plain Text	